

EVO52702517

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Application Program Interface for Network Software  
Platform**

Inventor(s):  
Brad Abrams

ATTORNEY'S DOCKET NO. MS1-861USC1

1 **RELATED APPLICATIONS**

2 This application is a continuation of application no. 09/902,936 entitled  
3 "Application Program Interface for Network Software Platform", filed July 10,  
4 2001 (now abandoned).

5 This relates to the following six patents, all of which are incorporated  
6 herein by reference:

7 U.S. Patent No. \_\_\_\_\_, entitled "Application Program  
8 Interface for Network Software Platform", which issued \_\_\_\_\_ from  
9 application Serial No. 09/902,811, filed July 10, 2001. (Attorney's Docket No.  
10 MS1-862US)

11 U.S. Patent No. \_\_\_\_\_, entitled "Application Program  
12 Interface for Network Software Platform", which issued \_\_\_\_\_ from  
13 application Serial No. 09/902,809, filed July 10, 2001. (Attorney's Docket No.  
14 MS1-863US)

15 U.S. Patent No. \_\_\_\_\_, entitled "Application Program  
16 Interface for Network Software Platform", which issued \_\_\_\_\_ from  
17 application Serial No. 09/902,560, filed July 10, 2001. (Attorney's Docket No.  
18 MS1-864US)

19 U.S. Patent No. \_\_\_\_\_, entitled "Application Program  
20 Interface for Network Software Platform", which issued \_\_\_\_\_ from  
21 application Serial No. 09/902,810, filed July 10, 2001. (Attorney's Docket No.  
22 MS1-865US)

23 U.S. Patent No. \_\_\_\_\_, entitled "Application Program  
24 Interface for Network Software Platform", which issued \_\_\_\_\_ from  
25

1 application Serial No. 09/902,812, filed July 10, 2001. (Attorney's Docket No.  
2 MS1-866US)

3 U.S. Patent No. \_\_\_\_\_, entitled "Application Program  
4 Interface for Network Software Platform", which issued \_\_\_\_\_ from  
5 application Serial No. 09/901,555, filed July 10, 2001. (Attorney's Docket No.  
6 MS1-867US)

### 7 8 **TECHNICAL FIELD**

9 This invention relates to network software, such as Web applications, and to  
10 computer software development of such network software. More particularly, this  
11 invention relates to an application program interface (API) that facilitates use of a  
12 network software platform by application programs and computer hardware.

### 13 14 **BACKGROUND**

15 Very early on, computer software came to be categorized as "operating  
16 system" software or "application" software. Broadly speaking, an application is  
17 software meant to perform a specific task for the computer user such as solving a  
18 mathematical equation or supporting word processing. The operating system is  
19 the software that manages and controls the computer hardware. The goal of the  
20 operating system is to make the computer resources available to the application  
21 programmer while at the same time, hiding the complexity necessary to actually  
22 control the hardware.

23 The operating system makes the resources available via functions that are  
24 collectively known as the Application Program Interface or API. The term API is  
25 also used in reference to a single one of these functions. The functions are often

1 grouped in terms of what resource or service they provide to the application  
2 programmer. Application software requests resources by calling individual API  
3 functions. API functions also serve as the means by which messages and  
4 information provided by the operating system are relayed back to the application  
5 software.

6 In addition to changes in hardware, another factor driving the evolution of  
7 operating system software has been the desire to simplify and speed application  
8 software development. Application software development can be a daunting task,  
9 sometimes requiring years of developer time to create a sophisticated program  
10 with millions of lines of code. For a popular operating system such as Microsoft  
11 Windows®, application software developers write thousands of different  
12 applications each year that utilize the operating system. A coherent and usable  
13 operating system base is required to support so many diverse application  
14 developers.

15 Often, development of application software can be made simpler by making  
16 the operating system more complex. That is, if a function may be useful to several  
17 different application programs, it may be better to write it once for inclusion in the  
18 operating system, than requiring dozens of software developers to write it dozens  
19 of times for inclusion in dozens of different applications. In this manner, if the  
20 operating system supports a wide range of common functionality required by a  
21 number of applications, significant savings in applications software development  
22 costs and time can be achieved.

23 Regardless of where the line between operating system and application  
24 software is drawn, it is clear that for a useful operating system, the API between  
25

1 the operating system and the computer hardware and application software is as  
2 important as efficient internal operation of the operating system itself.

3 Over the past few years, the universal adoption of the Internet, and  
4 networking technology in general, has changed the landscape for computer  
5 software developers. Traditionally, software developers focused on single-site  
6 software applications for standalone desktop computers, or LAN-based computers  
7 that were connected to a limited number of other computers via a local area  
8 network (LAN). Such software applications were typically referred to as "shrink  
9 wrapped" products because the software was marketed and sold in a shrink-  
10 wrapped package. The applications utilized well-defined APIs to access the  
11 underlying operating system of the computer.

12 As the Internet evolved and gained widespread acceptance, the industry  
13 began to recognize the power of hosting applications at various sites on the World  
14 Wide Web (or simply the "Web"). In the networked world, clients from anywhere  
15 could submit requests to server-based applications hosted at diverse locations and  
16 receive responses back in fractions of a second. These Web applications, however,  
17 were typically developed using the same operating system platform that was  
18 originally developed for standalone computing machines or locally networked  
19 computers. Unfortunately, in some instances, these applications do not adequately  
20 transfer to the distributed computing regime. The underlying platform was simply  
21 not constructed with the idea of supporting limitless numbers of interconnected  
22 computers.

23 To accommodate the shift to the distributed computing environment being  
24 ushered in by the Internet, Microsoft Corporation is developing a network  
25 software platform known as the ".NET" platform (read as "Dot Net"). The

1 platform allows developers to create Web services that will execute over the  
2 Internet. Such a dynamic shift requires a new ground-up design of an entirely new  
3 API.

4 In response to this challenge, the inventors developed a unique set of API  
5 functions for Microsoft's .NET™ platform.

## 6 7 **SUMMARY**

8 An application program interface (API) provides a set of functions for  
9 application developers who build Web applications on a network platform, such as  
10 Microsoft Corporation's .NET™ platform.

## 11 12 **BRIEF DESCRIPTION OF THE DRAWINGS**

13 The same numbers are used throughout the drawings to reference like  
14 features.

15 Fig. 1 illustrates a network architecture in which clients access Web  
16 services over the Internet using conventional protocols.

17 Fig. 2 is a block diagram of a software architecture for Microsoft's .NET™  
18 platform, which includes an application program interface (API).

19 Fig. 3 is a block diagram of unique namespaces supported by the API, as  
20 well as function classes of the various API functions.

21 Fig. 4 is a block diagram of an exemplary computer that may execute all or  
22 part of the software architecture.

## 23 24 **BRIEF DESCRIPTION OF ACCOMPANYING COMPACT DISC**

25

Accompanying this specification is a duplicative set of compact discs, identified as "Copy 1" and "Copy 2". Each disc stores a compiled HTML help file identifying the API (application program interface) for Microsoft's .NET™ network platform. The file is named "cpref.chm" and was created on June 8, 2001. It is 30.81 Mbytes in size. The file can be executed on a Windows®-based computing device (e.g., IBM-PC, or equivalent) that executes a Windows®-brand operating system (e.g., Windows® NT, Windows® 98, Windows® 2000, etc.). The compiled HTML help file stored on the compact disc is hereby incorporated by reference. The compact disc itself is a CD-ROM, and conforms to the ISO 9660 standard.

Additionally, each compact disc stores 94 separate text files named "NamespaceName.txt" which contain the APIs listed in the compiled HTML help file. The text files comply with the ASCII format and may be read using a Windows®-based computing device (e.g., IBM-PC, or equivalent) that executes a Windows®-brand operating system (e.g., Windows® NT, Windows® 98, Windows® 2000, etc.). The text files stored on the compact disc are hereby incorporated by reference.

|  |           |          |
|--|-----------|----------|
| System.Windows.Forms.txt                       | 2,463,923 | 7/6/2001 |
| System.CodeDom.Compiler.txt                    | 163,205   | 7/6/2001 |
| System.ComponentModel.Design.txt               | 229,762   | 7/6/2001 |
| System.Configuration.Assemblies.txt            | 6,457     | 7/6/2001 |
| System.ComponentModel.txt                      | 534,420   | 7/6/2001 |
| System.ComponentModel.Design.Serialization.txt | 56,951    | 7/6/2001 |
| System.Configuration.txt                       | 24,160    | 7/6/2001 |
| System.txt                                     | 1,372,604 | 7/6/2001 |
| System.Net.txt                                 | 284,291   | 7/6/2001 |
| System.Collections.txt                         | 177,639   | 7/6/2001 |
| System.Globalization.txt                       | 331,753   | 7/6/2001 |
| System.Net.Sockets.txt                         | 137,612   | 7/6/2001 |
| System.Collections.Specialized.txt             | 99,154    | 7/6/2001 |
| System.Xml.Schema.txt                          | 122,405   | 7/6/2001 |

|    |  |         |          |
|----|--|---------|----------|
| 1  | System.Xml.Serialization.txt                       | 224,452 | 7/6/2001 |
|    | System.Xml.XPath.txt                               | 56,553  | 7/6/2001 |
|    | System.Xml.txt                                     | 416,578 | 7/6/2001 |
| 2  | System.Xml.Xsl.txt                                 | 3,552   | 7/6/2001 |
|    | System.Data.Common.txt                             | 114,227 | 7/7/2001 |
| 3  | System.Data.OleDb.txt                              | 155,509 | 7/7/2001 |
|    | System.Data.SqlClient.txt                          | 121,455 | 7/7/2001 |
| 4  | System.Data.SqlTypes.txt                           | 352,862 | 7/7/2001 |
|    | System.Diagnostics.txt                             | 399,479 | 7/7/2001 |
| 5  | System.DirectoryServices.txt                       | 98,856  | 7/7/2001 |
|    | System.Drawing.Design.txt                          | 89,887  | 7/7/2001 |
| 6  | System.Drawing.Drawing2D.txt                       | 212,421 | 7/7/2001 |
|    | System.Reflection.txt                              | 298,065 | 7/7/2001 |
| 7  | System.Drawing.txt                                 | 702,023 | 7/7/2001 |
|    | System.Drawing.Imaging.txt                         | 232,591 | 7/7/2001 |
| 8  | System.Drawing.Printing.txt                        | 142,134 | 7/7/2001 |
|    | System.Drawing.Text.txt                            | 8,501   | 7/7/2001 |
| 9  | System.EnterpriseServices.txt                      | 138,609 | 7/8/2001 |
|    | System.IO.txt                                      | 308,389 | 7/7/2001 |
| 10 | System.Resources.txt                               | 70,121  | 7/7/2001 |
|    | System.IO.IsolatedStorage.txt                      | 56,779  | 7/7/2001 |
| 11 | System.Messaging.txt                               | 342,690 | 7/7/2001 |
|    | System.Reflection.Emit.txt                         | 352,613 | 7/7/2001 |
| 12 | System.Runtime.CompilerServices.txt                | 20,020  | 7/7/2001 |
|    | System.Runtime.InteropServices.Expando.txt         | 1,497   | 7/7/2001 |
| 13 | System.Runtime.InteropServices.txt                 | 389,509 | 7/7/2001 |
|    | System.Runtime.Remoting.Activation.txt             | 12,595  | 7/7/2001 |
| 14 | System.Runtime.Remoting.Channels.txt               | 116,351 | 7/7/2001 |
|    | System.Runtime.Remoting.Channels.Http.txt          | 36,192  | 7/7/2001 |
| 15 | System.Runtime.Remoting.Channels.Tcp.txt           | 24,032  | 7/7/2001 |
|    | System.Runtime.Remoting.Contexts.txt               | 43,554  | 7/7/2001 |
| 16 | System.Runtime.Remoting.txt                        | 112,724 | 7/7/2001 |
|    | System.Runtime.Remoting.Lifetime.txt               | 14,216  | 7/7/2001 |
| 17 | System.Runtime.Remoting.Messaging.txt              | 69,733  | 7/7/2001 |
|    | System.Runtime.Remoting.Metadata.txt               | 10,824  | 7/7/2001 |
| 18 | System.Runtime.Remoting.Metadata.W3cXsd2001.txt    | 58,782  | 7/7/2001 |
|    | System.Runtime.Remoting.MetadataServices.txt       | 15,165  | 7/7/2001 |
| 19 | System.Runtime.Remoting.Proxies.txt                | 12,034  | 7/7/2001 |
|    | System.Runtime.Remoting.Services.txt               | 13,600  | 7/7/2001 |
| 20 | System.Runtime.Serialization.Formatters.Binary.txt | 9,694   | 7/7/2001 |
|    | System.Runtime.Serialization.Formatters.txt        | 16,288  | 7/7/2001 |
| 21 | System.Runtime.Serialization.txt                   | 122,559 | 7/7/2001 |
|    | System.Runtime.Serialization.Formatters.Soap.txt   | 9,712   | 7/7/2001 |
| 22 | System.Security.Cryptography.txt                   | 250,786 | 7/7/2001 |
|    | System.Security.Cryptography.X509Certificates.txt  | 26,564  | 7/7/2001 |
| 23 | System.Configuration.Install.txt                   | 57,485  | 7/8/2001 |
|    | System.Security.Permissions.txt                    | 206,364 | 7/7/2001 |
| 24 | System.Security.txt                                | 83,229  | 7/7/2001 |
|    | System.Security.Policy.txt                         | 162,414 | 7/7/2001 |
| 25 | System.Text.txt                                    | 130,394 | 7/7/2001 |
|    | System.Security.Principal.txt                      | 27,479  | 7/7/2001 |



|    |   |         |          |
|----|---|---------|----------|
| 1  | System.ServiceProcess.txt                                 | 77,072  | 7/7/2001 |
|    | System.Text.RegularExpressions.txt                        | 76,478  | 7/7/2001 |
|    | System.Threading.txt                                      | 111,902 | 7/7/2001 |
| 2  | System.Timers.txt   | 10,381  | 7/7/2001 |
|    | System.Windows.Forms.Design.txt                           | 168,099 | 7/7/2001 |
| 3  | System.Web.txt  | 237,045 | 7/9/2001 |
|    | System.Diagnostics.SymbolStore.txt                        | 51,472  | 7/8/2001 |
| 4  | System.Management.txt                                     | 255,522 | 7/8/2001 |
|    | System.Management.Instrumentation.txt                     | 14,199  | 7/8/2001 |
| 5  | System.Web.Caching.txt                                    | 26,389  | 7/9/2001 |
|    | System.Web.Configuration.txt                              | 7,820   | 7/9/2001 |
| 6  | System.Web.Hosting.txt                                    | 13,234  | 7/9/2001 |
|    | System.Web.Mail.txt                                       | 11,187  | 7/9/2001 |
| 7  | System.Web.Security.txt                                   | 77,598  | 7/9/2001 |
|    | System.Web.Services.txt                                   | 20,955  | 7/9/2001 |
| 8  | System.Web.Services.Configuration.txt                     | 8,242   | 7/9/2001 |
|    | System.Web.Services.Description.txt                       | 215,516 | 7/9/2001 |
| 9  | System.Web.Services.Discovery.txt                         | 80,061  | 7/9/2001 |
|    | System.Web.Services.Protocols.txt                         | 171,896 | 7/9/2001 |
| 10 | System.Web.SessionState.txt                               | 5,064   | 7/9/2001 |
|    | System.Web.UI.txt   | 254,142 | 7/9/2001 |
| 11 | System.Web.UI.Design.txt                                  | 120,182 | 7/9/2001 |
|    | System.Web.UI.Design.WebControls.txt                      | 77,222  | 7/9/2001 |
| 12 | System.Web.UI.HtmlControls.txt                            | 62,508  | 7/9/2001 |
|    | System.Web.UI.WebControls.txt                             | 607,903 | 7/9/2001 |
| 13 | System.CodeDom.txt  | 233,760 | 7/9/2001 |
|    | System.Data.txt   | 440,804 | 7/9/2001 |
| 14 | System.EnterpriseServices.CompensatingResourceManager.txt | 24,077  | 7/9/2001 |
|    | System.Security.Cryptography.Xml.txt                      | 86,585  | 7/9/2001 |

Also, each compact disc stores a file entitled "Common Language Runtime Specification" that contains the following Word® documents, their sizes, and date of creation:

|    |                             |           |           |
|----|-----------------------------|-----------|-----------|
| 20 | Document Format Information | 1KB       | 7/9/2001  |
|    | Glossary                    | 80KB      | 5/17/2001 |
| 21 | Partition I Architecture    | 3,350 KB  | 5/17/2001 |
|    | Partition II Metadata       | 10,494 KB | 5/17/2001 |
| 22 | Partition III CIL           | 1,107 KB  | 5/17/2001 |
|    | Partition IV Library        | 1,332 KB  | 5/17/2001 |
| 23 | Partition V Annexes         | 1,036 KB  | 5/17/2001 |
|    | WDO5-Review                 | 4,690 KB  | 5/17/2001 |

1 The Common Language Runtime Specification (inclusive of all documents above)  
2 is incorporated by reference.

## DETAILED DESCRIPTION

This disclosure addresses an application program interface (API) for a network platform upon which developers can build Web applications and services. More particularly, an exemplary API is described for the .NET™ platform created by Microsoft Corporation. The .NET™ platform is a software platform for Web services and Web applications implemented in the distributed computing environment. It represents the next generation of Internet computing, using open communication standards to communicate among loosely coupled Web services that are collaborating to perform a particular task.

In the described implementation, the .NET™ platform utilizes XML (extensible markup language), an open standard for describing data. XML is managed by the World Wide Web Consortium (W3C). XML is used for defining data elements on a Web page and business-to-business documents. XML uses a similar tag structure as HTML; however, whereas HTML defines how elements are displayed, XML defines what those elements contain. HTML uses predefined tags, but XML allows tags to be defined by the developer of the page. Thus, virtually any data items can be identified, allowing Web pages to function like database records. Through the use of XML and other open protocols, such as Simple Object Access Protocol (SOAP), the .NET™ platform allows integration of a wide range of services that can be tailored to the needs of the user. Although the embodiments described herein are described in conjunction with XML and other open standards, such are not required for the operation of the claimed invention. Other equally viable technologies will suffice to implement the inventions described herein.

1 As used herein, the phrase application program interface or API includes  
2 traditional interfaces that employ method or function calls, as well as remote calls  
3 (e.g., a proxy, stub relationship) and SOAP/XML invocations.

#### 5 EXEMPLARY NETWORK ENVIRONMENT

6 Fig. 1 shows a network environment 100 in which a network platform, such  
7 as the .NET™ platform, may be implemented. The network environment 100  
8 includes representative Web services 102(1), ..., 102(N), which provide services  
9 that can be accessed over a network 104 (e.g., Internet). The Web services,  
10 referenced generally as number 102, are programmable application components  
11 that are reusable and interact programmatically over the network 104, typically  
12 through industry standard Web protocols, such as XML, SOAP, WAP (wireless  
13 application protocol), HTTP (hypertext transport protocol), and SMTP (simple  
14 mail transfer protocol) although other means of interacting with the Web services  
15 over the network may also be used, such as Remote Procedure Call (RPC) or  
16 object broker type technology. A Web service can be self-describing and is often  
17 defined in terms of formats and ordering of messages.

18 Web services 102 are accessible directly by other services (as represented  
19 by communication link 106) or a software application, such as Web application  
20 110 (as represented by communication links 112 and 114). Each Web service 102  
21 is illustrated as including one or more servers that execute software to handle  
22 requests for particular services. Such services often maintain databases that store  
23 information to be served back to requesters. Web services may be configured to  
24 perform any one of a variety of different services. Examples of Web services  
25 include login verification, notification, database storage, stock quoting, location

1 directories, mapping, music, electronic wallet, calendar/scheduler, telephone  
2 listings, news and information, games, ticketing, and so on. The Web services can  
3 be combined with each other and with other applications to build intelligent  
4 interactive experiences.

5 The network environment 100 also includes representative client devices  
6 120(1), 120(2), 120(3), 120(4), ..., 120(M) that utilize the Web services 102 (as  
7 represented by communication link 122) and/or the Web application 110 (as  
8 represented by communication links 124, 126, and 128). The clients may  
9 communicate with one another using standard protocols as well, as represented by  
10 an exemplary XML link 130 between clients 120(3) and 120(4).

11 The client devices, referenced generally as number 120, can be  
12 implemented many different ways. Examples of possible client implementations  
13 include, without limitation, portable computers, stationary computers, tablet PCs,  
14 televisions/set-top boxes, wireless communication devices, personal digital  
15 assistants, gaming consoles, printers, photocopiers, and other smart devices.

16 The Web application 110 is an application designed to run on the network  
17 platform and may utilize the Web services 102 when handling and servicing  
18 requests from clients 120. The Web application 110 is composed of one or more  
19 software applications 130 that run atop a programming framework 132, which are  
20 executing on one or more servers 134 or other computer systems. Note that a  
21 portion of Web application 110 may actually reside on one or more of clients 120.  
22 Alternatively, Web application 110 may coordinate with other software on clients  
23 120 to actually accomplish its tasks.

24 The programming framework 132 is the structure that supports the  
25 applications and services developed by application developers. It permits multi-

1 language development and seamless integration by supporting multiple languages.  
2 It supports open protocols, such as SOAP, and encapsulates the underlying  
3 operating system and object model services. The framework provides a robust and  
4 secure execution environment for the multiple programming languages and offers  
5 secure, integrated class libraries.

6 The framework 132 is a multi-tiered architecture that includes an  
7 application program interface (API) layer 142, a common language runtime (CLR)  
8 layer 144, and an operating system/services layer 146. This layered architecture  
9 allows updates and modifications to various layers without impacting other  
10 portions of the framework. A common language specification (CLS) 140 allows  
11 designers of various languages to write code that is able to access underlying  
12 library functionality. The specification 140 functions as a contract between  
13 language designers and library designers that can be used to promote language  
14 interoperability. By adhering to the CLS, libraries written in one language can be  
15 directly accessible to code modules written in other languages to achieve seamless  
16 integration between code modules written in one language and code modules  
17 written in another language. One exemplary detailed implementation of a CLS is  
18 described in an ECMA standard created by participants in ECMA TC39/TG3.  
19 The reader is directed to the ECMA web site at [www.ecma.ch](http://www.ecma.ch).

20 The API layer 142 presents groups of functions that the applications 130  
21 can call to access the resources and services provided by layer 146. By exposing  
22 the API functions for a network platform, application developers can create Web  
23 applications for distributed computing systems that make full use of the network  
24 resources and other Web services, without needing to understand the complex  
25 interworkings of how those network resources actually operate or are made

1 available. Moreover, the Web applications can be written in any number of  
2 programming languages, and translated into an intermediate language supported  
3 by the common language runtime 144 and included as part of the common  
4 language specification 140. . In this way, the API layer 142 can provide methods  
5 for a wide and diverse variety of applications.

6 Additionally, the framework 132 can be configured to support API calls  
7 placed by remote applications executing remotely from the servers 134 that host  
8 the framework. Representative applications 148(1) and 148(2) residing on clients  
9 120(3) and 120(M), respectively, can use the API functions by making calls  
10 directly, or indirectly, to the API layer 142 over the network 104.

11 The framework may also be implemented at the clients. Client 120(3)  
12 represents the situation where a framework 150 is implemented at the client. This  
13 framework may be identical to server-based framework 132, or modified for client  
14 purposes. Alternatively, the client-based framework may be condensed in the  
15 event that the client is a limited or dedicated function device, such as a cellular  
16 phone, personal digital assistant, handheld computer, or other  
17 communication/computing device.

## 18 19 DEVELOPERS' PROGRAMMING FRAMEWORK

20 Fig. 2 shows the programming framework 132 in more detail. The  
21 common language specification (CLS) layer 140 supports applications written in a  
22 variety of languages 130(1), 130(2), 130(3), 130(4), ..., 130(K). Such application  
23 languages include Visual Basic, C++, C#, COBOL, Jscript, Perl, Eiffel, Python,  
24 and so on. The common language specification 140 specifies a subset of features  
25 or rules about features that, if followed, allow the various languages to

communicate. For example, some languages do not support a given type (e.g., an “int\*” type) that might otherwise be supported by the common language runtime 144. In this case, the common language specification 140 does not include the type. On the other hand, types that are supported by all or most languages (e.g., the “int[]” type) is included in common language specification 140 so library developers are free to use it and are assured that the languages can handle it. This ability to communicate results in seamless integration between code modules written in one language and code modules written in another language. Since different languages are particularly well suited to particular tasks, the seamless integration between languages allows a developer to select a particular language for a particular code module with the ability to use that code module with modules written in different languages. The common language runtime 144 allow seamless multi-language development, with cross language inheritance, and provide a robust and secure execution environment for the multiple programming languages. For more information on the common language specification 140 and the common language runtime 144, the reader is directed to co-pending applications entitled “Method and System for Compiling Multiple Languages”, filed 6/21/2000 (serial number 09/598,105) and “Unified Data Type System and Method” filed 7/10/2000 (serial number 09/613,289), which are incorporated by reference.

The framework 132 encapsulates the operating system 146(1) (e.g., Windows®-brand operating systems) and object model services 146(2) (e.g., Component Object Model (COM) or Distributed COM). The operating system 146(1) provides conventional functions, such as file management, notification, event handling, user interfaces (e.g., windowing, menus, dialogs, etc.), security, authentication, verification, processes and threads, memory management, and so



on. The object model services 146(2) provide interfacing with other objects to perform various tasks. Calls made to the API layer 142 are handed to the common language runtime layer 144 for local execution by the operating system 146(1) and/or object model services 146(2).

The API 142 groups API functions into multiple namespaces. Namespaces essentially define a collection of classes, interfaces, delegates, enumerations, and structures, which are collectively called “types”, that provide a specific set of related functionality. A class represents managed heap allocated data that has reference assignment semantics. A delegate is an object oriented function pointer. An enumeration is a special kind of value type that represents named constants. A structure represents static allocated data that has value assignment semantics. An interface defines a contract that other types can implement.

By using namespaces, a designer can organize a set of types into a hierarchical namespace. The designer is able to create multiple groups from the set of types, with each group containing at least one type that exposes logically related functionality. In the exemplary implementation, the API 142 is organized into four root namespaces: a first namespace 200 for Web applications, a second namespace 202 for client applications, a third namespace 204 for data and XML, and a fourth namespace 206 for base class libraries (BCLs). Each group can then be assigned a name. For instance, types in the Web applications namespace 200 are assigned the name “Web”, and types in the data and XML namespace 204 can be assigned names “Data” and “XML” respectively. The named groups can be organized under a single “global root” namespace for system level APIs, such as an overall System namespace. By selecting and prefixing a top level identifier, the types in each group can be easily referenced by a hierarchical name that includes

1 the selected top level identifier prefixed to the name of the group containing the  
2 type. For instance, types in the Web applications namespace 200 can be  
3 referenced using the hierarchical name "System.Web". In this way, the individual  
4 namespaces 200, 202, 204, and 206 become major branches off of the System  
5 namespace and can carry a designation where the individual namespaces are  
6 prefixed with a designator, such as a "System." prefix.

7 The Web applications namespace 200 pertains to Web based functionality,  
8 such as dynamically generated Web pages (e.g., Microsoft's Active Server Pages  
9 (ASP)). It supplies types that enable browser/server communication. The client  
10 applications namespace 202 pertains to drawing and client side UI functionality.  
11 It supplies types that enable drawing of two-dimensional (2D), imaging, and  
12 printing, as well as the ability to construct window forms, menus, boxes, and so  
13 on.

14 The data and XML namespace 204 relates to connectivity to data sources  
15 and XML functionality. It supplies classes, interfaces, delegates, and  
16 enumerations that enable security, specify data types, and serialize objects into  
17 XML format documents or streams. The base class libraries (BCL) namespace  
18 206 pertains to basic system and runtime functionality. It contains the  
19 fundamental types and base classes that define commonly-used value and  
20 reference data types, events and event handlers, interfaces, attributes, and  
21 processing exceptions.

22 In addition to the framework 132, programming tools 210 are provided to  
23 assist the developer in building Web services and/or applications. One example of  
24 the programming tools 200 is Visual Studio™, a multi-language suite of  
25 programming tools offered by Microsoft Corporation.

## ROOT API NAMESPACES

Fig. 3 shows the API 142 and its four root namespaces in more detail. In one embodiment, the namespaces are identified according to a hierarchical naming convention in which strings of names are concatenated with periods. For instance, the Web applications namespace 200 is identified by the root name "System.Web". Within the "System.Web" namespace is another namespace for Web services, identified as "System.Web.Services", which further identifies another namespace for a description known as "System.Web.Services.Description". With this naming convention in mind, the following provides a general overview of selected namespaces of the API 142, although other naming conventions could be used with equal effect.

The Web applications namespace 200 ("System.Web") defines additional namespaces, including:

- A services namespace 300 ("System.Web.Services") containing classes that enable a developer to build and use Web services. The services namespace 300 defines additional namespaces, including a description namespace 302 ("System.Web.Services.Description") containing classes that enable a developer to publicly describe a Web service via a service description language (such as WSDL, a specification available from the W3C), a discovery namespace 304 ("System.Web.Services.Discovery") containing classes that allow Web service consumers to locate available Web Services on a Web server, and a protocols namespace 306

1 (“System.Web.Services.Protocols”) containing classes that define  
2 the protocols used to transmit data across a network during  
3 communication between Web service clients and the Web service  
4 itself.

- 5 • A caching namespace 308 (“System.Web.Caching”) containing  
6 classes that enable developers to decrease Web application response  
7 time through temporarily caching frequently used resources on the  
8 server. This includes ASP.NET pages, web services, and user  
9 controls. (ASP.NET is the updated version of Microsoft’s ASP  
10 technology.) Additionally, a cache dictionary is available for  
11 developers to store frequently used resources, such as hash tables  
12 and other data structures.
- 13 • A configuration namespace 310 (“System.Web.Configuration”) containing classes that are used to read configuration data in for an application.
- 14 • A UI namespace 312 (“System.Web.UI”) containing types that allow  
15 developers to create controls and pages that will appear in Web  
16 applications as user interfaces on a Web page. This namespace  
17 includes the control class, which provides all web based controls,  
18 whether those encapsulating HTML elements, higher level Web  
19 controls, or even custom User controls, with a common set of  
20 functionality. Also provided are classes which provide the web  
21 forms server controls data binding functionality, the ability to save  
22 the view state of a given control or page, as well as parsing  
23 functionality for both programmable and literal controls. Within the  
24  
25

1 UI namespace 312 are two additional namespaces: an HTML  
2 controls namespace 314 (“System.Web.UI.HtmlControls”)  
3 containing classes that permit developers to interact with types that  
4 encapsulates html 3.2 elements create HTML controls, and a Web  
5 controls namespace 316 (“System.Web.UI.WebControls”)  
6 containing classes that allow developers to create higher level Web  
7 controls.

- 8 • A security namespace 318 (“System.Web.Security”) containing  
9 classes used to implement security in web server applications, such  
10 as basic authentication, challenge response authentication, and role  
11 based authentication.
- 12 • A session state namespace 320 (“System.Web.SessionState”)  
13 containing classes used to access session state values (i.e., data that  
14 lives across requests for the lifetime of the session) as well as  
15 session-level settings and lifetime management methods.

16  
17 The client applications namespace 202 is composed of two namespaces:

- 18  
19 • A windows forms namespace 322 (“System.Windows.Forms”)  
20 containing classes for creating Windows®-based client applications  
21 that take full advantage of the rich user interface features available in  
22 the Microsoft Windows® operating system, such as the ability to  
23 drag and drop screen elements. Such classes may include wrapped  
24 APIs available in the Microsoft Windows® operating system that  
25 are used in a windowing UI environment. Within this namespace

are a design namespace 324 (“System.Windows.Forms.Design”) that contains classes to extend design-time support for Windows forms and a component model namespace 326 (“System.Windows.Forms.ComponentModel”) that contains the windows form implementation of the general component model defined in System.ComponentModel. This namespace contains designer tools, such as Visual Studio, which offer a rich experience for developers at design time.

- A drawing namespace 328 (“System.Drawing”) containing classes for graphics functionality. The drawing namespace 328 includes a 2D drawing namespace 330 (“System.Drawing.Drawing2D”) that contains classes and enumerations to provide advanced 2-dimensional and vector graphics functionality, an imaging namespace 332 (“System.Drawing.Imaging”) that contains classes for advanced imaging functionality, a printing namespace 334 (“System.Drawing.Printing”) that contains classes to permit developers to customize printing, and a text namespace 336 (“System.Drawing.Text”) that contains classes for advanced typography functionality.

The data and XML namespace 204 is composed of two namespaces:

- A data namespace 340 (“System.Data”) containing classes that enable developers to build components that efficiently manage data from multiple data sources. It implements an architecture that, in a

1 disconnected scenario (such as the Internet), provides tools to  
2 request, update, and reconcile data in multiple tier systems. The data  
3 namespace 340 includes a common namespace 342 that contains  
4 types shared by data providers. A data provider describes a  
5 collection of types used to access a data source, such as a database,  
6 in the managed space. The data namespace 340 also includes an  
7 OLE DB namespace 344 that contains types pertaining to data used  
8 in object-oriented databases (e.g., Microsoft's SQL Server), and a  
9 SQL client namespace 346 that contains types pertaining to data  
10 used by SQL clients. The data namespace also includes a SQL types  
11 namespace 348 ("System.Data.SqlTypes") that contains classes for  
12 native data types within Microsoft's SQL Server. The classes  
13 provide a safer, faster alternative to other data types. Using the  
14 objects within this namespace helps prevent type conversion errors  
15 caused in situations where loss of precision could occur. Because  
16 other data types are converted to and from SQL types behind the  
17 scenes, explicitly creating and using objects within this namespace  
18 results in faster code as well.

- 19 • An XML namespace 350 ("System.XML") containing classes that  
20 provide standards-based support for processing XML. The supported  
21 standards include XML (e.g., version 1.0), XML Namespaces (both  
22 stream level and DOM), XML Schemas, XPath expressions, XSL/T  
23 transformations, DOM Level 2 Core, and SOAP (e.g., version 1.1).  
24 The XML namespace 350 includes an XSLT namespace 352  
25 ("System.XML.Xsl") that contains classes and enumerations to

1 support XSLT (Extensible Stylesheet Language Transformations),  
2 an Xpath namespace 354 (“System.XML.Xpath”) that contains an  
3 XPath parser and evaluation engine, and a serialization namespace  
4 356 (“System.XML.Serialization”) that contains classes used to  
5 serialize objects into XML format documents or streams.

6  
7 The base class library namespace 206 (“System”) includes the following  
8 namespaces:

- 9
- 10 • A collections namespace 360 (“System.Collections”) containing  
11 interfaces and classes that define various collections of objects, such  
12 as lists, queues, arrays, hash tables and dictionaries.
  - 13 • A configuration namespace 362 (“System.Configuration”) containing  
14 classes and interfaces that allow developers to  
15 programmatically access configuration settings and handle errors in  
16 configuration files.
  - 17 • A diagnostics namespace 364 (“System.Diagnostics”) containing  
18 classes that are used to debug applications and to trace code  
19 execution. The namespace allows developers to start system  
20 processes, read and write to event logs, and monitor system  
21 performance using performance counters.
  - 22 • A globalization namespace 366 (“System.Globalization”) containing  
23 classes that define culture-related information, including the  
24 language, the country/region, the calendars in use, the format  
25



1 patterns for dates, currency and numbers, and the sort order for  
2 strings.

- 3 • An I/O namespace 368 (“System.IO”) containing the infrastructure  
4 pieces to operate with the input/output of data streams, files, and  
5 directories. This namespace includes a model for working with  
6 streams of bytes, higher level readers and writers which consume  
7 those bytes, various constructions or implementations of the streams  
8 (e.g., FileStream and MemoryStream) and, a set of utility classes for  
9 working with files and directories.
- 10 • A net namespace 370 (“System.Net”) providing an extensive set of  
11 classes for building network-enabled application, referred to as the  
12 Net Class Libraries (NCL). One element to the design of the Net  
13 Class Libraries is an extensible, layered approach to exposing  
14 networking functionality. The NCL stack contains three basic  
15 layers. A base layer (System.Net.Socket) provides access to an  
16 interface to TCP/IP, the communications protocol of UNIX networks  
17 and the Internet. One example of such an interface is the “WinSock  
18 API” from Microsoft Corporation. The next layer is the Transport  
19 Protocol classes, which support such transport protocols as TCP and  
20 UDP. Developers may write their own protocol classes to provide  
21 support for protocols such as IGMP and ICMP. The third layer is  
22 the Web request, which provides an abstract factory pattern for the  
23 creation of other protocol classes. The NCL provides  
24 implementations for Hyper Text Transport Protocol (HTTP).

- A reflection namespace (“System.Reflection”) 372 containing types that provide a managed view of loaded types, methods, and fields, with the ability to dynamically create and invoke types.
- A resources namespace 374 (“System.Resources”) containing classes and interfaces that allow developers to create, store and manage various culture-specific resources used in an application.
- A security namespace 376 (“System.Security”) supporting the underlying structure of the security system, including interfaces, attributes, exceptions, and base classes for permissions.
- A service process namespace 378 (“System.ServiceProcess”) containing classes that allow developers to install and run services. Services are long-running executables that run without a user interface. They can be installed to run under a system account that enables them to be started at computer reboot. Services whose implementation is derived from processing in one class can define specific behavior for start, stop, pause, and continue commands, as well as behavior to take when the system shuts down.
- A text namespace 380 (“System.Text”) containing classes representing various types of encodings (e.g., ASCII, Unicode, UTF-7, and UTF-8), abstract base classes for converting blocks of characters to and from blocks of bytes, and a helper class that manipulates and formats string objects without creating intermediate instances.
- A threading namespace 382 (“System.Threading”) containing classes and interfaces that enable multi-threaded programming. The

1 threading namespace includes a ThreadPool class that manages  
2 groups of threads, a Timer class that enables a delegate to be called  
3 after a specified amount of time, and a Mutex class for  
4 synchronizing mutually-exclusive threads. This namespace also  
5 provides classes for thread scheduling, wait notification, and  
6 deadlock resolution.

- 7 • A runtime namespace 384 (“System.Runtime”) containing multiple  
8 namespaces concerning runtime features, including an interoperation  
9 services namespace 386 (“System.Runtime.InteropServices”) that  
10 contains a collection of classes useful for accessing COM objects.  
11 The types in the InteropServices namespace fall into the following  
12 areas of functionality: attributes, exceptions, managed definitions of  
13 COM types, wrappers, type converters, and the Marshal class. The  
14 runtime namespace 384 further includes a remoting namespace 388  
15 (“System.Runtime.Remoting”) that contains classes and interfaces  
16 allowing developers to create and configure distributed applications.  
17 Another namespace within the runtime namespace 384 is a  
18 serialization namespace 390 (“System.Runtime.Serialization”) that  
19 contains classes used for serializing and deserializing objects.  
20 Serialization is the process of converting an object or a graph of  
21 objects into a linear sequence of bytes for either storage or  
22 transmission to another location.

## EXEMPLARY COMPUTING SYSTEM AND ENVIRONMENT

Fig. 4 illustrates an example of a suitable computing environment 400 within which the programming framework 132 may be implemented (either fully or partially). The computing environment 400 may be utilized in the computer and network architectures described herein.

The exemplary computing environment 400 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computing environment 400 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary computing environment 400.

The framework 132 may be implemented with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, multiprocessor systems, microprocessor-based systems, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and so on. Compact or subset versions of the framework may also be implemented in clients of limited resources, such as cellular phones, personal digital assistants, handheld computers, or other communication/computing devices.

The framework 132 may be described in the general context of computer-executable instructions, such as program modules, being executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks

1 or implement particular abstract data types. The framework 132 may also be  
2 practiced in distributed computing environments where tasks are performed by  
3 remote processing devices that are linked through a communications network. In  
4 a distributed computing environment, program modules may be located in both  
5 local and remote computer storage media including memory storage devices.

6 The computing environment 400 includes a general-purpose computing  
7 device in the form of a computer 402. The components of computer 402 can  
8 include, by are not limited to, one or more processors or processing units 404, a  
9 system memory 406, and a system bus 408 that couples various system  
10 components including the processor 404 to the system memory 406.

11 The system bus 408 represents one or more of several possible types of bus  
12 structures, including a memory bus or memory controller, a peripheral bus, an  
13 accelerated graphics port, and a processor or local bus using any of a variety of  
14 bus architectures. By way of example, such architectures can include an Industry  
15 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an  
16 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)  
17 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a  
18 Mezzanine bus.

19 Computer 402 typically includes a variety of computer readable media.  
20 Such media can be any available media that is accessible by computer 402 and  
21 includes both volatile and non-volatile media, removable and non-removable  
22 media.

23 The system memory 406 includes computer readable media in the form of  
24 volatile memory, such as random access memory (RAM) 410, and/or non-volatile  
25 memory, such as read only memory (ROM) 412. A basic input/output system

1 (BIOS) 414, containing the basic routines that help to transfer information  
2 between elements within computer 402, such as during start-up, is stored in ROM  
3 412. RAM 410 typically contains data and/or program modules that are  
4 immediately accessible to and/or presently operated on by the processing unit 404.

5 Computer 402 may also include other removable/non-removable,  
6 volatile/non-volatile computer storage media. By way of example, Fig. 4  
7 illustrates a hard disk drive 416 for reading from and writing to a non-removable,  
8 non-volatile magnetic media (not shown), a magnetic disk drive 418 for reading  
9 from and writing to a removable, non-volatile magnetic disk 420 (e.g., a "floppy  
10 disk"), and an optical disk drive 422 for reading from and/or writing to a  
11 removable, non-volatile optical disk 424 such as a CD-ROM, DVD-ROM, or other  
12 optical media. The hard disk drive 416, magnetic disk drive 418, and optical disk  
13 drive 422 are each connected to the system bus 408 by one or more data media  
14 interfaces 426. Alternatively, the hard disk drive 416, magnetic disk drive 418,  
15 and optical disk drive 422 can be connected to the system bus 408 by one or more  
16 interfaces (not shown).

17 The disk drives and their associated computer-readable media provide non-  
18 volatile storage of computer readable instructions, data structures, program  
19 modules, and other data for computer 402. Although the example illustrates a hard  
20 disk 416, a removable magnetic disk 420, and a removable optical disk 424, it is to  
21 be appreciated that other types of computer readable media which can store data  
22 that is accessible by a computer, such as magnetic cassettes or other magnetic  
23 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or  
24 other optical storage, random access memories (RAM), read only memories  
25 (ROM), electrically erasable programmable read-only memory (EEPROM), and

1 the like, can also be utilized to implement the exemplary computing system and  
2 environment.

3 Any number of program modules can be stored on the hard disk 416,  
4 magnetic disk 420, optical disk 424, ROM 412, and/or RAM 410, including by  
5 way of example, an operating system 426, one or more application programs 428,  
6 other program modules 430, and program data 432. Each of the operating system  
7 426, one or more application programs 428, other program modules 430, and  
8 program data 432 (or some combination thereof) may include elements of the  
9 programming framework 132.

10 A user can enter commands and information into computer 402 via input  
11 devices such as a keyboard 434 and a pointing device 436 (e.g., a "mouse").  
12 Other input devices 438 (not shown specifically) may include a microphone,  
13 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and  
14 other input devices are connected to the processing unit 404 via input/output  
15 interfaces 440 that are coupled to the system bus 408, but may be connected by  
16 other interface and bus structures, such as a parallel port, game port, or a universal  
17 serial bus (USB).

18 A monitor 442 or other type of display device can also be connected to the  
19 system bus 408 via an interface, such as a video adapter 444. In addition to the  
20 monitor 442, other output peripheral devices can include components such as  
21 speakers (not shown) and a printer 446 which can be connected to computer 402  
22 via the input/output interfaces 440.

23 Computer 402 can operate in a networked environment using logical  
24 connections to one or more remote computers, such as a remote computing device  
25 448. By way of example, the remote computing device 448 can be a personal

1 computer, portable computer, a server, a router, a network computer, a peer device  
2 or other common network node, and so on. The remote computing device 448 is  
3 illustrated as a portable computer that can include many or all of the elements and  
4 features described herein relative to computer 402.

5 Logical connections between computer 402 and the remote computer 448  
6 are depicted as a local area network (LAN) 450 and a general wide area network  
7 (WAN) 452. Such networking environments are commonplace in offices,  
8 enterprise-wide computer networks, intranets, and the Internet.

9 When implemented in a LAN networking environment, the computer 402 is  
10 connected to a local network 450 via a network interface or adapter 454. When  
11 implemented in a WAN networking environment, the computer 402 typically  
12 includes a modem 456 or other means for establishing communications over the  
13 wide network 452. The modem 456, which can be internal or external to computer  
14 402, can be connected to the system bus 408 via the input/output interfaces 440 or  
15 other appropriate mechanisms. It is to be appreciated that the illustrated network  
16 connections are exemplary and that other means of establishing communication  
17 link(s) between the computers 402 and 448 can be employed.

18 In a networked environment, such as that illustrated with computing  
19 environment 400, program modules depicted relative to the computer 402, or  
20 portions thereof, may be stored in a remote memory storage device. By way of  
21 example, remote application programs 458 reside on a memory device of remote  
22 computer 448. For purposes of illustration, application programs and other  
23 executable program components such as the operating system are illustrated herein  
24 as discrete blocks, although it is recognized that such programs and components  
25



1 reside at various times in different storage components of the computing device  
2 402, and are executed by the data processor(s) of the computer.

3 An implementation of the framework 132, and particularly, the API 142 or  
4 calls made to the API 142, may be stored on or transmitted across some form of  
5 computer readable media. Computer readable media can be any available media  
6 that can be accessed by a computer. By way of example, and not limitation,  
7 computer readable media may comprise "computer storage media" and  
8 "communications media." "Computer storage media" include volatile and non-  
9 volatile, removable and non-removable media implemented in any method or  
10 technology for storage of information such as computer readable instructions, data  
11 structures, program modules, or other data. Computer storage media includes, but  
12 is not limited to, RAM, ROM, EEPROM, flash memory or other memory  
13 technology, CD-ROM, digital versatile disks (DVD) or other optical storage,  
14 magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage  
15 devices, or any other medium which can be used to store the desired information  
16 and which can be accessed by a computer.

17 "Communication media" typically embodies computer readable  
18 instructions, data structures, program modules, or other data in a modulated data  
19 signal, such as carrier wave or other transport mechanism. Communication media  
20 also includes any information delivery media. The term "modulated data signal"  
21 means a signal that has one or more of its characteristics set or changed in such a  
22 manner as to encode information in the signal. By way of example, and not  
23 limitation, communication media includes wired media such as a wired network or  
24 direct-wired connection, and wireless media such as acoustic, RF, infrared, and  
25

1 other wireless media. Combinations of any of the above are also included within  
2 the scope of computer readable media.

3 Alternatively, portions of the framework may be implemented in hardware  
4 or a combination of hardware, software, and/or firmware. For example, one or  
5 more application specific integrated circuits (ASICs) or programmable logic  
6 devices (PLDs) could be designed or programmed to implement one or more  
7 portions of the framework.

### 8 9 **Conclusion**

10 Although the invention has been described in language specific to structural  
11 features and/or methodological acts, it is to be understood that the invention  
12 defined in the appended claims is not necessarily limited to the specific features or  
13 acts described. Rather, the specific features and acts are disclosed as exemplary  
14 forms of implementing the claimed invention.